monoxgboost

jaume cloquell capo

6 de febrero de 2019

Clasificación monotónica

```
#'@function createTraintAndTestPartition
#'@description General las particiones de test y train
#'
#'@param dataset Dataset original
#'@param SplitRatio Porcentaje de valores en train y test
#'@return Una lista con las dos particiones
createTraintAndTestPartition <- function (dataset, SplitRatio = 0.75) {
  set.seed(123)
  dt <- list()
  sample = sample.split(dataset,SplitRatio = SplitRatio)
  dt[["train"]] = subset(dataset,sample ==TRUE)
  dt[["test"]] = subset(dataset, sample==FALSE)
  return (dt)
}

#'@function transformToBinary
#'@description Modificacion del datatset con valores binarios en la columna de salida por cada clase
#'
#'@param data data.frame de entrada
#'@param clase clase de la variable de salida
#'@return Dataset modificado
transformToBinary <- function(i, data){
  n <- ncol(data)
  #se cambia la ?tima columna distinguiendo si se es mayor o menor que la clase i
  data[,n] <- as.ordered(ifelse(data[,n]>i, 1, 0))
  data[,n] <- as.factor(data[,n])
  return(data)
}

#'@function makePrediction
#'@description Genera un modelo xgBoost
#'
#'@param data Dataset de train para generar el modelo
makePrediction <- function (data) {
  xgboost(data.matrix(data[, -ncol(data)]), label = as.numeric(data[,ncol(data)])-1,
          nrounds = 2, params = list(monotone_constraints = 1, objective = "binary:logistic"), verbose =
}
```

```
#'@function monotomicClassification
#'@description Aplicación de un modelo de clasificación con OVA y xgBoost
#'
#'@param data data.frame de entrada. Se espera la variable de salida en la última columna
#'@return Procentaje de acierto.
monotomicClassification <- function(dataset){
```

```
  data <- createTraintAndTestPartition(dataset)
  testData <- data[["test"]]
  trainData <- data[["train"]]

  num.column <- ncol(trainData)

  x = 1
  for(i in 1:length(unique(trainData[,num.column]))-1)
  {
    binaryTrainData <- transformToBinary(i, trainData)

    modelo <- makePrediction(binaryTrainData)

    x = x + ifelse(predict(modelo, as.matrix(testData[, -num.column]))>0.5,1,0)
  }

  return(sum(x==testData[,num.column])/nrow(testData))
}

esl <- read.arff("esl.arff")
era <- read.arff("era.arff")
lev <- read.arff("lev.arff")
swd <- read.arff("swd.arff")
monotomicClassification(esl)
```

```
## [1] 0.7061856
```

```
monotomicClassification(era)
```

```
## [1] 0.2775
```

```
monotomicClassification(lev)
```

```
## [1] 0.1725
```

```
monotomicClassification(swd)
```

```
## [1] 0.1764706
```